1997

# The Concept of Error in a Database: An Application of Temporal Databases

Gautam Bhargava
*Iowa State University*

Shashi K. Gadia
*Iowa State University*

# The Concept of Error in a Database: An Application of Temporal Databases

**Abstract**
The existing database models do not capture the difference between updates intended to make changes and corrections. The information about errors is external to the database and such information cannot be queried. We give a model to capture the concept of error in a database. The model consisting of 2-dimensional temporal relations is a consistent extension of the classical relational model as well as our l-dimensional temporal relational model. To circumvent the identity of an object from becoming corrupt due to the presence of errors, we make a copy of the correct identity and permanently glue (anchor) it to the object. The transition from the l-dimensional case to the 2-dimensional case is complex, but most of this complexity is absorbed by the system and not passed on to the user. This paper is a promising application of temporal databases to main stream databases.

**Disciplines**
Databases and Information Systems | Systems Architecture

# THE CONCEPT OF ERROR IN A DATABASE:
## An application of temporal databases[1]

Gautam Bhargava[2] and Shashi K. Gadia
Computer Science Department
Iowa State University
Ames, IA 50011, USA

gadia@atanasoff.cs.iastate.edu

**Abstract.** The existing database models do not capture the difference between updates intended to make changes and corrections. The information about errors is external to the database and such information cannot be queried. We give a model to capture the concept of error in a database. The model consisting of 2–dimensional temporal relations is a consistent extension of the classical relational model as well as our 1–dimensional temporal relational model. To circumvent the identity of an object from becoming corrupt due to the presence of errors, we make a copy of the correct identity and permanently glue (anchor) it to the object. The transition from the 1–dimensional case to the 2–dimensional case is complex, but most of this complexity is absorbed by the system and not passed on to the user. This paper is a promising application of temporal databases to main stream databases.

### 1. INTRODUCTION.

Conventional database systems only record the latest version of reality. Any errors made in the recording of data can be corrected, so that the latest version is correct, but no record of erroneous information can be maintained. In this paper we present a relational model that allows us to not only record all versions of data, but also to query them for errors.

Temporal relations can store multiple versions of object histories by associating timestamps with data values. These timestamps can be given different interpretations, leading to different semantic representations. For example, if the timestamps are thought of as the real world time, the temporal data can be viewed as the real world history of the objects under consideration. Another approach is to view the timestamps as transaction time, which leads to the data being viewed as the database history of objects under consideration.

In this paper we present a 2–dimensional temporal model and show how an SQL–like query language, called SQL/2 may be used for querying the model. This allows us to formulate and answer queries that ask, among other things, for the past errors made in recording

---

[1] This work was supported in part by the National Science Foundation, USA, under grant IRI- 8810704.

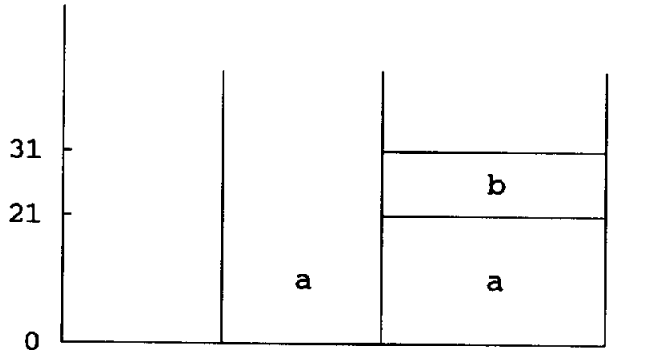
[2] Current address: Santa Teresa Laboratory, IBM Corporation, San Jose, CA 95141, USA

information in the database. As we do not require any synchronicity in the happening of an event and its recording in the database, we associate a 2-dimensional timestamp with every data value – one dimension, called the real world time, allows us to assemble a version of history of an object, and the other dimension, called the transaction time, allows us to revise our knowledge of a version of history.

1.1. Example. Figure 1.1 shows the tabular and pictorial representations of a 2–dimensional data value. It says that according to our knowledge during [40,80], the value was going to be a forever; but at 81 our knowledge changed in that the value was a during [10,20] in the real world, b during [21,30] in the real world, and the object ceased to exist at the instant 31 in the real world. In this paper we will consider our knowledge at NOW (the current time) to be correct. By this convention, during [40,80] our knowledge was not always correct; e.g. we thought that the value during [21,30] in the real world was a, whereas in reality it was b.

| | | | |
|---|---|---|---|
| [40,80] | × | [0,∞) | a |
| [81,NOW] | × | [0,20] | a |
| | × | [21,30] | b |

(a) Tabular representation




(b) Pictorial representation

Figure 1.1. Representations of a 2–dimensional data value

1.2. The concept of anchor. The 2–dimensional case causes an interesting problem, which has to be overcome before a reasonable query language for it can be given. In classical and historical databases we believe all data values are correct, but in the 2–dimensional case correct and incorrect values co–exist. The relational database support a value oriented paradigm; in the classical and historical databases a value, being correct, can identify itself. However, this convenience is not available in the 2–dimensional case. In this paper we will assume that our

current knowledge of objects is correct. What we need is a mechanism to identify our total knowledge by our current knowledge. This is done by introducing the concept of an anchor. A copy of the current knowledge is stored in an anchor, which is then glued to the total knowledge. The anchor is a sturdy copy of the identity, and it is not destructible. The following example illustrates the concept of an anchor.

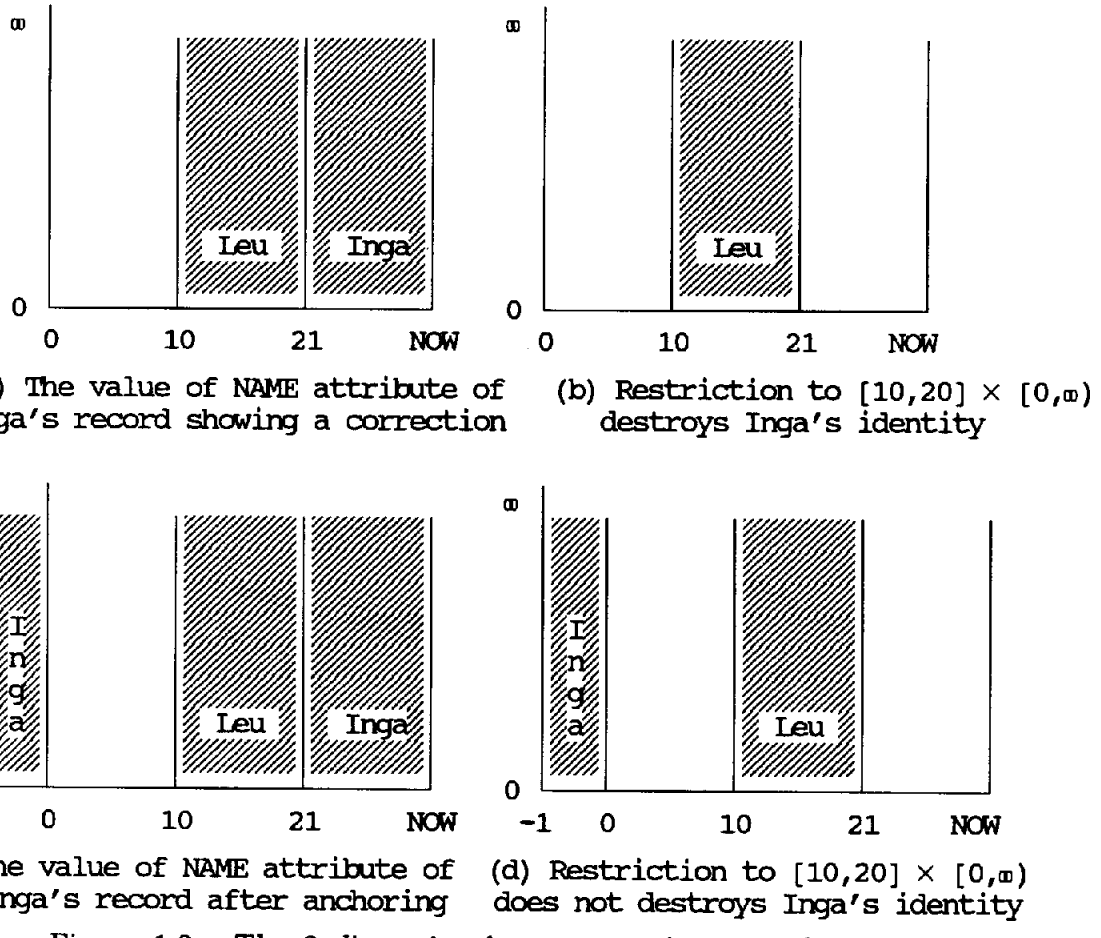



Figure 1.2. The 2-dimensional case necessitates anchoring

1.2.1. Example. Consider an employee relation emp over NAME SALARY DEPT. We suppose there is a record of Inga in the emp relation, but her name was erroneously recorded as Leu. At transaction time 21 it was corrected to Inga. Informally, suppose we compute an expression of the form $\sigma$(emp$\restriction$[10,20]×[0,∞); NAME = Inga), where we are first required to compute emp$\restriction$[10,20]×[0,∞), the emp relation with all its attribute values restricted to the time domain [10,20]×[0,∞), and then retrieve Inga's record. Figure 1.2 (b) shows the part of Inga's NAME attribute that remains after it is restricted to [10,20]×[0,∞). Clearly, at this point there is no way of knowing that it is a part of Inga's record. This means we cannot proceed with our

calculations without misinterpreting the data. The solution is to make the identity of Inga indestructible before any operations are performed on it. This is done by attaching an anchor to the attribute value, as shown in Figure 1.2 (c). We have chosen to store the anchor at an abstract instant -1 for ease of drawing diagrams. The anchor is nothing but the current version of the history. Figure 1.2 (d) shows how the identity of Inga is not destroyed as it is saved in the anchor.

1.3. User friendliness of the algebra. Anchoring of a relation is automatically done by the system when a user queries it. If $\xi$ is an anchored attribute value, $\lceil\xi$ denotes the anchor of $\xi$. The construct $\lceil\xi$ is "read-only," making the anchor available to a user for query purposes, without giving him/her the capability of destroying it. Using the anchor the query of example 1.2.1 can be expressed as $\sigma(\text{emp}\restriction[10,20]\times[0,\infty);\ \lceil\text{NAME} = \text{Inga})$ to retrieve the intended result.

In the real world, where a concept of absolute correctness is not available, it is customary to assume the current knowledge to be correct; our convention of using the current knowledge as the anchor mimics this aspect of the real world. As we have seen, the concept of an anchor is necessary to let the meaning of information persist beyond an application of an arbitrary number of algebraic operators. At this point it is worthwhile to mention an interesting phenomenon. The emp relation in Figure 3.1 shows records of Inga and Leu. It is motivated by the scenario that at some point it is realized that their identities were confused with each other, and in order to correct this situation a correction was made in both records simultaneously. Without an anchor the query of Example 1.2.1, $\sigma(\text{emp}\restriction[10,20]\times[0,\infty);\ \text{NAME} = \text{Inga})$, may retrieve Leu's record, instead of Inga's.

In Section 2 we discuss the 1-dimensional case. We introduce 1-dimensional historical relations, keys, weak relations [Ga1,Ga2] and weak keys. Essentially, a set of attributes is a weak key if it is a key of all snapshots. In the 1-dimensional case we see that a weak key renders identity to objects (Theorem 1); however, unlike for classical snapshot relation, the structure of a historical relation depends upon the choice of the key. In the 2-dimensional case we need more work to obtain a counterpart of Theorem 1. This is discussed next.

1.4. Transaction units and extraneous information. In Section 3 we generalize the one dimensional model to the 2-dimensional case. We find that in porting Theorem 1 to the 2-dimensional case we face several problems, making the generalization somewhat difficult. The first problem is about the fragility of the identity of an object due to presence of errors. As discussed above, this problem is solved by using anchors. The second problem is that if we allow arbitrary weak relations, we may end up splicing a part of one object to a part of another object. To stop this we introduce the concept of a transaction unit, the database history of an

object at a specified instant in the real world (see Figure 3.2(d)), and require that it be split across tuples. The third problem is that the extraneous portion of the history of an object, that is the portion which exists in the database, when the object does not exist in the real world, cannot be covered with data values, and this information becomes identity-less under a change of a key or restructuring due to other operations. Eventually, Theorem 2, the 2-dimensional counterpart of Theorem 1, is obtained. In Section 4 we introduce primitives for querying for errors. We give examples to show how to query errors using relational algebra and an SQL-like language.

1.5. Related works. Algebras for (1-dimensional) historical relations have appeared in [Ga1-Ga2,GV,GY2,LJ,Ta,TG]. [Ga1,Ga2,GV] mainly deal with weak temporal relations. [Ga3,GY1] show how use of temporal elements make a substantial simplification in query languages for temporal databases. The 2-dimensional concept of time was introduced in [Sn]. [Sa1,Sa2,NA] give models and SQL-like query languages for a temporal database. [BG] views a temporal relation as the database history of objects using the transaction time rather than the real world time. In that model one has the ability to restore the updates and results of queries executed in the past, and thus it is termed a zero information-loss model. [GY2] is a very preliminary version of this paper, and does not include any of the complexity associated with the 2-dimensional case.

1.6. The homogeneity assumption. The homogeneity requirement on our relations states that the time domain for all attributes in a tuple should be the same. Almost all researchers (except [Ta,TG]) use homogeneity implicitly without explicitly mentioning it. If one wishes, the homogeneity requirement can be circumvented by generalizing it; such an exercise is beyond the scope of this paper.

## 2. THE 1-DIMENSIONAL HISTORICAL RELATIONS.

In this section we assume that the real world time universe $[0,\infty) = \{0, 1, 2, \cdots\}$ is given.

2.1. Temporal elements. A *temporal element* is a finite union of intervals. For example, $[0,20] \cup [30,\infty)$ is a temporal element. Temporal elements are closed under $\cup$, $\cap$ and $\neg$ (complementation) and form a Boolean algebra; using them as timestamps instead of intervals or instants simplifies a user interface substantially [Ga3,GY1].

2.2. Temporal assignments. To capture time varying nature of the real world, we informally define a *temporal assignment* $\xi$ to an attribute A, with a temporal element $\mu$ as its domain to be a step-function from $\mu$ into dom(A) such that the range of $\xi$ is finite. (Formally, we require the

range of $\xi$, denoted $|\xi|$ to be finite, and the inverse image of a under $\xi$ to be a temporal element for every a in $|\xi|$.) The domain of $\xi$ is denoted as $[\![\xi]\!]$. A *tuple* $\tau$ over R is a function such that $\tau(A)$ is an assignment. We assume that the tuples are *homogeneous*; i.e., the time domain within a tuple does not vary from one attribute to another. $\xi\restriction\mu$ denotes the restriction of $\xi$ to a temporal element $\mu$ as a function.

2.3. $\theta$-navigation. The temporal counterpart of A$\theta$B of classical databases is $[\![\xi_1 \theta \xi_2]\!] = \{t \in [\![\xi_1]\!] \cap [\![\xi_2]\!]: \xi_1(t)$ is in $\theta$ relationship with $\xi_2(t)\}$. Whereas A$\theta$B evaluates to TRUE or FALSE, $[\![\xi_1 \theta \xi_2]\!]$ evaluates to a temporal element which lies between $\emptyset$ and $[\![\xi_1]\!] \cap [\![\xi_2]\!]$. If the value is $\emptyset$, it says that the two assignments $\xi_1$ and $\xi_2$ were never related.

2.4. Objects, identity and relations. Suppose $\tau$ and $\tau'$ are tuples over R and $K \subseteq R$. We say that $\tau$ is K-*unique* if $\tau(A)$ does not change with time for all $A \in K$. We say that $\tau$ and $\tau'$ *agree* on K, if they are K-unique and take the same value for all $A \in K$. A *historical relation* r over R with $K \subseteq R$ as its key is a set of tuples such that (i) every $\tau \in r$ is K-unique, and (ii) $\tau$ and $\tau'$ agree implies $\tau = \tau'$ for all $\tau, \tau'$ in r.

Figure 2.1(a) shows $\texttt{flights}_1$ relation over FLIGHT FROM TO DEPARTS, with FLIGHT as its key. Figure 2.1(b) shows $\texttt{flights}_2$ relation over FLIGHT FROM TO DEPARTS, with FROM TO DEPARTS as its key.

2.5. Weak equality. Two relations r(R) and s(R) are said to be *weakly equal* [Ga2], written $r \sim s$, if their snapshots are the same at all instants of time.

Note that $\texttt{flight}_1$ and $\texttt{flight}_2$ of Figure 2.1 are different relations; they even have different number of tuples. But the two relations are weakly equal, as all their snapshots are the same (e.g. Figure 2.1.(c) shows the snapshot of the two relations at instant 25).

$K \subseteq R$ is said to be a *weak key* of r if the functional dependency $K \rightarrow R$ holds in every snapshot of r. Note that FLIGHT (also FROM TO DEPARTS) is a weak key of $\texttt{flights}_1$ as well as $\texttt{flights}_2$.

THEOREM 1. If r is a historical relation over R, and $K \subseteq R$. Then K is a weak key of r if and only if there is a unique historical relation $s \sim r$ with K as its key.

The relation s in the above theorem is denoted as $I_K(r)$, and $I_K$ is called a *weak identity operator*. The theorem says that to make (computed) relations meaningful, we only need to worry about their key.

2.6. An algebra. An interesting feature of $r \cup s$, $r - s$, $\Pi_X(r)$, $\sigma(r; f)$, and $r \bowtie s$, in the classical algebra is that they result into relations, and hence one expression can be nested inside another. We extend this feature to the temporal case to allow relations and temporal elements to be

computed. Temporal expressions are formed using constant temporal elements, $[\![A]\!]$, $[\![A\theta B]\!]$, $[\![r]\!]$ (the time domain of a relation r), $\cup$, $\cap$ and $\neg$, and Boolean expressions are formed using $A \subseteq B$, $\mu \subseteq \nu$, $\vee$, $\wedge$, and $\neg$, where $\mu$ and $\nu$ are temporal expressions.

| FLIGHT | FROM | TO | DEPARTS |
|---|---|---|---|
| [11,∞) F261 | [11,∞) JFK | [11,∞) CHI | [11,20] 4PM<br>[21,∞) 5PM |
| [11,∞) F361 | [11,∞) JFK | [11,∞) CHI | [11,20] 6PM<br>[21,∞) 4PM |
| [20,∞) F461 | [20,∞) DSM | [20,∞) LAX | [20,∞) 4PM |

(a) $\texttt{flights}_1$ with FLIGHT as the key

| FLIGHT | FROM | TO | DEPARTS |
|---|---|---|---|
| [11,20] F261<br>[21,∞) F361 | [11,∞) JFK | [11,∞) CHI | [11,∞) 4PM |
| [21,∞) F261 | [21,∞) JFK | [21,∞) CHI | [21,∞) 5PM |
| [11,20] F361 | [11,20] JFK | [11,20] CHI | [11,20] 6PM |
| [20,∞) F461 | [20,∞) DSM | [20,∞) LAX | [20,∞) 4PM |

(b) $\texttt{flights}_2$ with FROM TO DEPARTS as the key

| FLIGHT | FROM | TO | DEPARTS |
|---|---|---|---|
| F261 | JFK | CHI | 5PM |
| F361 | JFK | CHI | 4PM |
| F461 | DSM | LAX | 4PM |

(c) snapshot of $\texttt{flights}_1$ and $\texttt{flights}_2$ at instant 25

Figure 2.1. Flights information as relations with different keys

2.6.1. Relational expressions. The expression $I_K(r)$, derived from Theorem 1 plays an important role. For example, if K is the key of r(R) and s(R), one may not be able to form r∪s with the same key. If this is anticipated, the user should carefully choose a key K', and compute $I_{K'}(r) \cup I_{K'}(s)$. Note that this computation will always succeed with K'=R, which may be used as a default. r−s is without any such problem. r⋈s is defined in a manner that it results in a

homogeneous relation; a key has to be designated for it by the user, the union of the keys of r and s may be used as a default. The problem with $\Pi_X(r)$ is that the key of r may not be included in the projected attributes X; the syntactic form $\Pi_X(r; K')$ allowing a user to input the intended key K' is more appropriate.

| NAME | SALARY | DEPT |
|---|---|---|
| [0,4] John | [0,1] 25K<br>[2,4] 30K | [0,2] Toys<br>[3,4] Shoes |

$emp_1$ with NAME as its key

| NAME | SALARY | DEPT |
|---|---|---|
| [0,1] John | [0,1] 25K | [0,1] Toys |
| [2,2] John | [2,2] 30K | [2,2] Toys |
| [3,4] John | [3,4] 30K | [3,4] Shoes |

$emp_2$ with NAME DEPT SALARY as key

(a) Weakly equal relations $emp_1$ and $emp_2$

| NAME | SALARY | DEPT |
|---|---|---|
| [0,2] John | [0,1] 25K<br>[1,2] 30K | [0,2] Toys |

$\sigma(emp_1)$

| NAME | SALARY | DEPT |
|---|---|---|

(empty relation)

$\sigma(emp_2)$

(b) $\sigma(emp_1)$ and $\sigma(emp_2)$ are not weakly equal

Figure 2.2. $\sigma$ is not weakly invariant

2.6.2. Non-weak invariance of $\sigma$. Perhaps the most powerful operator is $\sigma$. It is an interesting operator in temporal databases as it not weakly invariant, i.e., r~s does not imply $\sigma(r) \sim \sigma(s)$. Thus this operator cannot be captured in an algebra for weak relations given in [Ga1,Ga2]. In the classical case, this operator has the form $\sigma(r; f) = \{\tau: \tau \in r \wedge f(\tau)\}$. For the historical case it is $\sigma(r\restriction\mu; f)$, where f is a Boolean expression, and $\mu$ is a temporal expression, and evaluates to $\{\tau\restriction\mu(\tau): \tau \in r \wedge f(\tau)\}$, where $\tau\restriction\mu(\tau)$ allows one to select the restriction of $\tau$ to the temporal element $\mu(\tau)$, whose value depends upon $\tau$.

2.6.2.1. Example. The query expression $\sigma(emp\restriction[\![DEPT = Toys]\!]; [\![SALARY \geq 25K]\!] \supseteq [1,3])$ asks for information about employees while they were in Toys department provided they earned a salary of at least 25K during the time [1,3]. In Figure 2.2 (a) we consider a state of an employee relation; on the left we show $emp_1$ with NAME as its key, and on right we show $emp_2$ with NAME SALARY DEPT as its key. In part (b) we show $\sigma(emp_1)$ and $\sigma(emp_2)$. Clearly, the two forms $emp_1$ and $emp_2$ are weakly equal, they only differ in their choice of key. But $\sigma(emp_1)$ and $\sigma(emp_2)$ are not weakly equal.

This example also indicates the importance of time stamping at the attribute level. If time stamps are applied at the tuple level, the only way to express the employee relation is $emp_2$ of Figure 2.2 (a). This means that if we use tuple timestamps, we may either miss important operators, or would have to define them in a less intuitive manner.

## 3. THE 2-DIMENSIONAL ANCHORED RELATIONS.

In the classical and historical models, the values are assumed to be correct, and so by default they identify the objects they reside in. In a model for changing knowledge, this situation does not hold. We assume that our current knowledge of the real world is correct and expect it to provide identity to the total knowledge.

| NAME | SALARY | DEPT |
|---|---|---|
| [ 8,59] × [11, ∞) John<br>[60,NOW] × [11,60] John | [ 8,52] × [11, ∞) 15K<br>[53,54] × [11,49] 15K<br>[50, ∞) 20K<br>[55,59] × [11,49] 15K<br>[50,54] 20K<br>[55, ∞) 25K<br>[60,NOW] × [11,49] 15K<br>[50,54] 20K<br>[55,60] 25K | [ 8,39] × [11, ∞) Toys<br>[40,59] × [11,44] Toys<br>[45, ∞) Shoes<br>[60,NOW] × [11,44] Toys<br>[45,60] Shoes |
| [10,20] × [0, ∞) Leu<br>[21,NOW] × [0, ∞) Inga | [10,NOW] × [0, ∞) 25K | [10,NOW] × [0, ∞) Clothing |
| [10,20] × [0, ∞) Inga<br>[21,NOW] × [0, ∞) Leu | [10,NOW] × [0, ∞) 23K | [10,NOW] × [0, ∞) Toys |

The emp 2-relation

| DEPT | MANAGER |
|---|---|
| [ 8,48] × [11, ∞) Toys<br>[49,NOW] × [11,49] Toys | [ 8,42] × [11, ∞) John<br>[43,44] × [11,44] John<br>[45,48] × [11,44] John<br>[45, ∞) Leu<br>[49,NOW] × [11,44] John<br>[45,49] Leu |

The management 2-relation

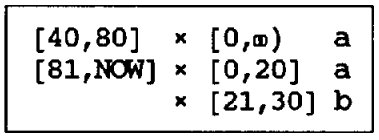
Figure 3.1. The personnel 2-database

The 2–dimensional universe of time is $[0,\text{NOW}]\times[0,\infty)$, where NOW denotes the current time. The second dimension is the *real world time*, and the first, called the *transaction time*, is used to model our changing knowledge. A *temporal element* is a finite union of rectangles; clearly, they form a Boolean algebra with $\cup$, $\cap$ and $\neg$. A *temporal assignment* $\xi$, $|\xi|$, $[\![\xi]\!]$, a *tuple*, *homogeneity*, *relation*, *weak equality* of relations, *key*, *weak key*, etc., from Section 2 are extended to the 2–dimensional case in a natural manner. Sometimes we use the prefix "2–" for the 2–dimensional structures.

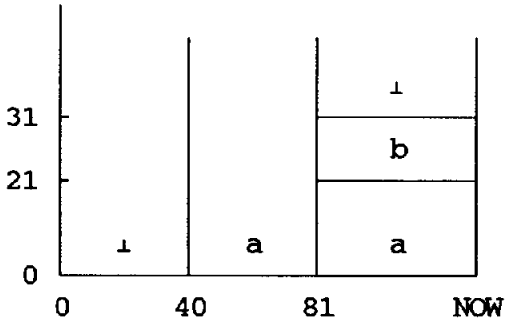
We consider an emp 2–relation over NAME SALARY DEPT with NAME as its weak key, and a management 2–relation over DEPT MANAGER with DEPT as its weak key. The database {emp,management} is called the personnel database. Figure 3.1 shows a state of the personnel database.

3.1. Identity and anchors. A temporal assignment in the 1–dimensional historical case consists of correct values, which by default identify themselves. A serious problem that arises in giving an algebra for the 2–dimensional case is that an algebraic operation may sometimes shrink an assignment, and the correct values my disappear from it. This amounts to loosing the identity of the assignment. (Also see Example 1.2.1 and Figure 1.2.) Our solution to this problem is to glue a copy of this identity to the assignment itself, and treat this copy as a non–destructible one. To do this we introduce an abstract transaction time instant –1, and store (anchor) the correct values along this instant. For an illustration consider the assignment $\xi$ of Figure 1.1 which, for the ease of reading, is also shown in Figure 3.2 (a) and (b). Figure 3.2(c) shows the result of anchoring the assignment $\xi$ of part (b), inspired by our assumption that the information at NOW is correct. A 2–dimensional relation after anchoring every assignment in it is called an *anchored relation*. A relation is automatically anchored at run time by the system.

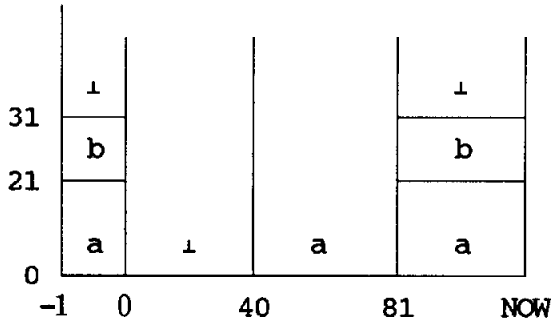
3.2. Restructuring, transaction units and extraneous information. In order to preserve the meaning of information, we cannot allow $\xi$ to be broken–up into arbitrarily small chunks under restructuring. A *transaction unit* is information contained in $\xi$ for a fixed real world time instant, see Figure 3.2(d). It is intuitively clear that under restructuring, this information should be treated as a unit, and only be allowed to migrate as a whole. Therefore we define two relations to be *weakly transaction equivalent* if they are weakly equal, and have the same transaction units. It is also clear that only the values in the anchor should determine the granularity of chunks for restructuring. Figure 3.2(e) shows how the assignment $\xi$ may be broken–up by a restructuring operator. However, one of the chunks looses its identity. *Extraneous information* in $\xi$ is the information existing in the database while the object does not exist in the real world according to the correct version. Thus the identity–less chunk in $\xi$



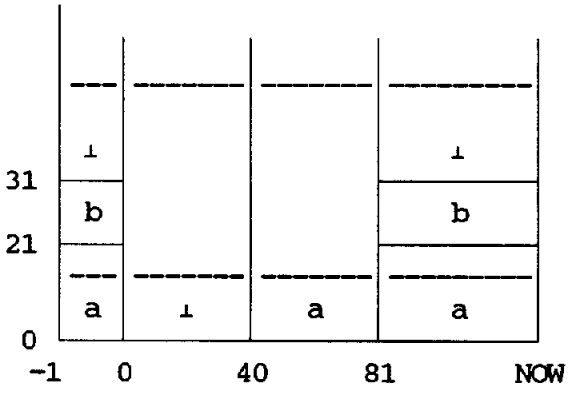

(a) A two dimensional assignment $\xi$



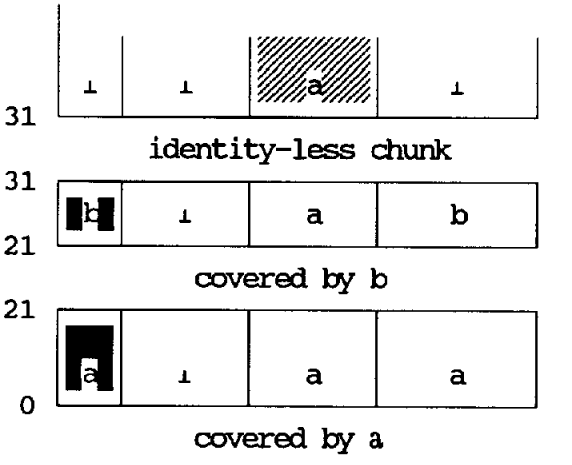

(b) Pictorial representation of $\xi$




(c) anchoring identity of $\xi$ to $\xi$




(d) ---- is a transaction unit, it should not be split during restructuring




(e) The value oriented decomposition $\xi$, ▨ is the extraneous information in $\xi$

Figure 3.1. Assignment $\xi$, its identity and value oriented decomposition

becomes extraneous information, and its destination in the restructuring is not clear. Perhaps the only logical alternative is to discard it. The information obtained from $\xi$ after discarding the extraneous information is denoted $\lfloor\xi$. $\lfloor r$ is obtained from r in a similar way. If r is an anchored relation over R, then $K \subseteq R$ is a *weak key* of r provided $K \rightarrow R$ holds in r. Finally, we obtain the following counterpart of Theorem 1, for the 2-dimensional case.

THEOREM 2. Suppose that r is an anchored relation and $K \subseteq R$. If $K \rightarrow R$ holds in r then there is a unique anchored relation s, such that s is weakly transaction equivalent to $\lfloor r$ with K as its key.

The relation s in the above theorem is denoted as $I_K(r)$, and $I_K$ is called a *weak identity operator for* anchored relations.

3.3. Algebra for the 2-dimensional case. The concept of an anchor plays an important role in providing a semantics for the algebraic operators. Its main use is to let the meaning of information persist beyond algebraic manipulation. Theorem 2 points to a potential problem. It says that only the $\lfloor r$ portion of r behaves properly; $r - \lfloor r$ is discarded whenever a restructuring becomes necessary. Fortunately, the most interesting operator, the selection $\sigma$, does not cause any restructuring, and so it does not discard any extraneous information. Due to lack of space we do not discuss the algebra any further, but in the next section we show how to use it to query errors.

## 4. ERRORS AND THEIR QUERY.

We have assumed that our current knowledge of the real world is correct. Thus, in order to determine errors in $\xi(t,t')$, we compare it with the correct value $\xi(\text{NOW},t')$, and classify errors as follows.

No error if $\xi(t,t') = \xi(\text{NOW},t')$,
Missing information if $(t,t') \notin [\![\xi]\!]$ and $(\text{NOW},t') \in [\![\xi]\!]$
Extraneous information if $(t,t') \in [\![\xi]\!]$ and $(\text{NOW},t') \notin [\![\xi]\!]$ and
Incorrect information if $\xi(t,t') \neq \xi(\text{NOW},t')$.

4.2. Primitives for querying errors. Corresponding to the above classification of errors, we introduce the following primitives in the algebra for the 2-dimensional model for querying errors.

$$\mathcal{E}_N(\xi) = \{(t,t'): \xi(t,t') = \xi(\text{NOW},t')\}$$
$$\mathcal{E}_M(\xi) = \{(t,t'): (t,t') \notin [\![\xi]\!] \text{ and } (\text{NOW},t') \in [\![\xi]\!]\}$$
$$\mathcal{E}_E(\xi) = \{(t,t'): \xi(t,t') \in [\![\xi]\!] \text{ and } \xi(\text{NOW},t') \notin [\![\xi]\!]\}$$

$$\mathcal{E}_I(\xi) = \{(t,t'): \xi(t,t') \neq \xi(\text{NOW},t')\}.$$
$$\mathcal{E}(\xi) = \mathcal{E}_M \cup \mathcal{E}_E \cup \mathcal{E}_I$$

4.3. Examples. An extensive framework already exists in our algebra for querying the 2-dimensional model. To query for errors, we merely have to use the primitives $\mathcal{E}_N$, $\mathcal{E}_M$, $\mathcal{E}_E$, $\mathcal{E}_I$ and $\mathcal{E}$ introduced above. Note that $\mathcal{E}_N$, $\mathcal{E}_M$, $\mathcal{E}_E$, $\mathcal{E}_I$ and $\mathcal{E}$ capture the temporal element during which there is no error, missing information, extraneous information, incorrect information, and some error, respectively. Their use is illustrated in the following examples. All the examples refer to the personnel database of Figure 3.1.

4.3.1. Example. "Give complete details about John when his salary value was incorrectly recorded in the database" is expressed as $\sigma(\text{emp}\restriction\mathcal{E}_I(\text{SALARY}); \lceil\text{NAME} = \text{John})$ in the relational algebra. Note that $\lceil$A refers to the anchor of A. In SQL/2, it is expressed as follows.

```
SELECT *
RESTRICTED TO E_I(SALARY)
FROM emp
WHERE ⌈NAME = John
```

4.3.2. Example. Note that the primitive for querying extraneous information in an attribute A is $\mathcal{E}_E$(A). Because of homogeneity, its value is independent of the choice of attribute A within a relation. The query "report extraneous information in the emp relation" is expressed in the relational algebra as $\sigma(\text{emp}\restriction\mathcal{E}_E; \text{TRUE})$. An SQL/2 expression is as follows.

```
SELECT *
RESTRICTED TO E_E
FROM emp
```

$\sigma(r\restriction\mu; f)$ is a powerful operator, as it allows an arbitrary level of nesting inside f and $\mu$; f and $\mu$ may involve relational or Boolean expression, and in turn those may involve other expressions, and so on. This helps us express queries using selection which sometimes may appear to be joins. The following example illustrates this point. We add an optional third argument $\nu$ to $\sigma$, to obtain $\sigma(r\restriction\mu; f; \nu)$ which restricts the real world time dimension of a retrieved tuple to a 1-dimensional temporal expression $\nu$. Although $\nu$ is computed on the basis of any information in an assignment $\xi$, our intention is to allow the the user to input the correct (anchored) information through it.

4.3.3. Example. "Give the time when there was an error in recording John's salary while he was (really) working in Toys" is expressed as $[\![\sigma(\text{emp}\restriction(\neg\mathcal{E}_N(\text{SALARY})); \lceil\text{NAME} = \text{John}; [\![\lceil\text{DEPT} =$

Toys]])]] in the relational algebra. The SQL/2 counterpart is as shown below; [[*]] retrieves the time domains, instead of the tuples, and the third argument of $\sigma$ is captured by the DURING clause.

```
SELECT [[*]]
RESTRICTED TO ¬E_N(SALARY)
DURING [[⌈DEPT = Toys]]
FROM emp
WHERE ⌈NAME = John
```

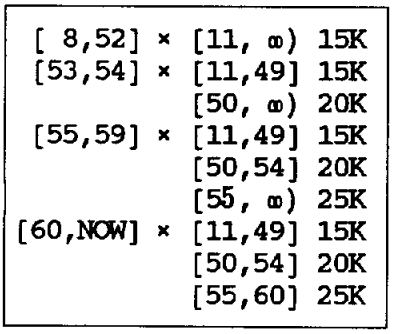
4.3.4. Example. "Give information about managers in Toys, during the times the event in example 4.3.3 happened." If we denote this expression in example 4.3.3. as $\mu$, then the complete query is written as $\sigma$(management$\restriction\mu$; ⌈DEPT = ⌈Toys). In SQL/2 it is expressed as follows.

```
SELECT MANAGER
RESTRICTED TO      (SELECT [[*]]
                    RESTRICTED TO ¬E_N(SALARY)
                    DURING [[⌈DEPT = Toys]]
                    FROM emp
                    WHERE ⌈NAME = John)
FROM management
WHERE ⌈DEPT = ⌈Toys
```

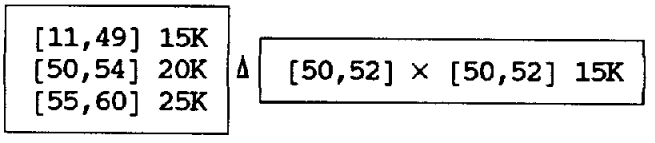
## 5. AVOIDING DATA REDUNDANCY.

There is substantial data redundancy in our two dimensional relations. This is particularly true if one is not interested in querying the errors arising from trying to predict the future. In this case $\{(t,t'): t' \leq t\}$, denoted as ◪, becomes the universe of time domains. Even though the time domains $\mu$ and $\nu$ may seem different in $[0,\text{NOW}] \times [0,\infty)$, we may disregard these differences $\mu \cap$ ◪ and $\nu \cap$ ◪ are the same. A 2-assignment $\xi$ may be denoted in "$\Delta$-notation" as (*the anchor of* $\xi$) $\Delta$ (*the deviation of* $\xi$ *from its anchor*). For example, John's salary from Figure 3.1(a) is shown in Figure 5.1(a). When the future is of no concern to us, this can be represented more compactly as shown in Figure 5.1(b).

Thus the storage requirements can be reduced to that of a historical relation + a quantity proportional to the "amount of error in the database." As the main purpose of the paper is to show the feasibility of a good model and a user friendly query language, we have not considered the storage requirements in details.




(a) The Salary value of John from Figure 3.1(a)




(b) A compact representation for the salary of John

Figure 5.1. Avoiding data redundancy

## 6. CONCLUSION AND RELATED WORK.

The main theme of the paper was to give a formal definition and query capability for the concept of an error in a database. Our framework is a consistent extension of the classical relational model. The 2-dimensional model can host a variety of users, by restricting their views to different parts of the 2-dimensional universe of time. [GB] gives a more comprehensive and a formal treatment of the 2-dimensional model, including an algebra for querying updates and errors; it also introduces a concept of user hierarchy. [BG] gives a formal treatment of transaction time and shows how to make updates in a database non-destructive by providing a framework where the effect, as well a transaction itself, can be restored at any time in the future. Such a system is appropriately called a zero information loss system. The ideas contained in [GB], [BG] and this paper inch toward building zero information loss systems of the future.

## REFERENCES

[BG] Bhargava, Gautam and Gadia, Shashi K. *Achieving zero information- loss in a classical database environment.* Proc. 15th International VLDB, August 1989.

[Ga1] Gadia, Shashi K. *A Homogeneous Relational Model and Query Languages for Temporal Databases.* ACM-Transactions on Database Systems, pp 418-448, vol 13, 1988.

[Ga2] Gadia, Shashi K. *Weak temporal relations.* Proc. Fifth ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1985.

[Ga3] Gadia, Shashi K. *The role of temporal elements in temporal databases.* Quarterly Bull. of the Computer Society of IEEE Technical Committee on Data Engineering, December, 1988.

[GB] Gadia, Shashi K. and Bhargava, Gautam. *A Formal Treatment of Updates and Errors in a Relational Database.* Submitted for publication.

[GV] Gadia, Shashi K. and Vaishnav, Jay. *A query language for a temporal database.* Proc. Fourth ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1985.

[GY1] Gadia, Shashi K. and Yeung, Chuen-Sing. *Inadequacy of Interval Timestamps in Temporal Databases.* To appear in Information Sciences.

[GY2] Gadia, Shashi K. and Yeung, Chuen-Sing. *A Generalized Model for a Relational Temporal Database.* Proc ACM-SIGMOD Int. Conf. on Management of Data, June 1988, pp. 251-257.

[LJ] Lorentzos, Nikos A. and Johnson, Roger G. *Extending relational algebra to manipulate temporal data.* Information Systems, Vol 13, 1988, pp 289-296.

[NA] Navathe, S.B. and Ahmed, Rafi. *A temporal relational model and query language for a temporal database.* Technical Report, 1986, Database Systems R&D Center, University of Florida.

[Sa1] Sarda, Nandlal L. *Algebra and query language for a historical data model.* The Computer journal, Vol 33, 1990, pp 11-18.

[Sa2] Sarda, Nandlal L. *Extension to SQL for historical databases.* IEEE Transactions on Knowledge and Data Engineering, Vol 2, 1990, pp 220-230.

[Sn] Snodgrass, Richard. *The Temporal Query Language TQuel.* ACM Transactions on Database Systems, Vol 12, 1987, pp 247-298.

[Ta] Tansel, A.U. *Adding Time Dimension to Relational Model Model and Extending Relational Algebra.* Information Systems, 1986.

[TG] Tansel, A.U. and Garnett, L. *Nested historical relations.* Proc ACM-SIGMOD Int. Conf. on Management of Data, 1989, pp. 284-293.